# Automated IP Management Python Project

## Project Description

Playing the role of a security professional working at a health care company, my job task is to update a file that identifies the employees who can access restricted content. The file “allow\_list.txt” contains information on who is able to work with personal patient records. Access to a separate network subnet that contains the restricted patient data is controlled by IP address. The allow list that permits access to the restricted subnetwork contains IP addresses for the users and is stored in the “allow\_list.txt” file. There is also a remove list that identifies which employees must be removed from the allow list.

My current task is to create an algorithm that uses Python code to check whether the allow list contains any IP addresses identified on the remove list. If any IP addresses in the restricted list are present in the allowed list, the file contents need to be updated by removing restricted user IPs from the allow list.

## Open File Containing Allow List

To begin the project, I opened the “allow\_list.txt” file by assigning the filename as a string to the “importFile” variable. I also manually defined the IP addresses to be removed as a list stored in the “removeList” variable. Note: The program assumes that the allow list is present in the same directory as the running Python program. Otherwise, the directory of the file must be included.

#imports the allow list

importFile = "allow\_list.txt"

#defines IP addresses to be removed.  
removeList = ["192.168.10.183", "192.168.2.63", "192.168.50.54"]

I then used a with statement to open the file.

#opens the allow list.

with open(importFile, "r") as file:

My algorithm uses with and the .open() function in read mode to open the allow list file for the purpose of reading it. The purpose of opening the file is to allow me to access the IP addresses stored in the allow list file. Using with allows efficient resource management by closing the file after exiting the statement. The code within the with open(importFile, “r”) as file statement has two parameters, “importFile”, which indicates the file being imported, and “r” which indicates I want to read the file. I also use an as statement to store the output into a file variable.

## Read File Contents

To read the file contents I used the .read() method, converting the contents into a string.

#stores the list as a string in a variable called “allowList”.

allowList = file.read()

The .read() function converts the file contents into a string, which is stored in the allowList variable. The resulting issue is I would like to automatically process the IP address in a for loop. While regex can accomplish finding IP addresses in the string, converting the contents of allowList into a list is more practical.

## Convert String to List

To accomplish this, I use the .split() method.

#converts the string into a list so it may be used in the upcoming loop.

allowList = allowList.split()

The .split() method is used by appending it to the end of a string and automatically converts a string to a list, making it easier to remove IP addresses from the allow list. The .split() method will automatically divide the IP addresses that are separated by spaces and, in this algorithm, will store them in the allowList variable.

## Iterate Through Remove List

To iterate through the remove list, I implemented a for loop.

#begins a loop to iterate through IPs

for ip in removeList:

The for loop takes IP addresses contained in the removeList and defines them in the ip variable. The loop will run until it has checked every IP address in the removeList. I now need to automate decision making by implemented the use of an if statement.

## Removing the IP Addresses

The integration of the if statement automatically processes data and performs decision making.

#if the IP is present, it will be removed.

if ip in allowList:  
 allowList.remove(ip)

In this case, the algorithm compares every IP in removeList and, if it is present, the program will remove the IP from the list stored in the allowList variable.

## Update Original Allowed IPs

The last step is to update the original allowList file.

#rebuilds the allow list.

allowList = "\n".join(allowList)

The “\n”.join(allowList) code ensures all the IPs are converted back into a string, separating each IP on a new line. The final step is to write the file however, since the with and open() methods close files after they are used, I need to reuse the methods with a few changes.

#writes the reformed allow list.

with open(importFile, "w") as file:   
 file.write(allowList)

I again run a with statement and open() function. Instead of using “r” as a parameter and the .read() method like before, I use “w” and .write() to write to the file.

## Results

The following images illustrates the text file before and after running the code.

A screenshot of a computer

Description automatically generated

Figure 1: Before Running Code

A screenshot of a computer

Description automatically generated

Figure 2: After Running Code

## Summary

The algorithm I created removes IP addresses found in a remove\_list variable from the “allow\_list.txt” file. The algorithm functions by opening the file, turning the IP addresses into a list, and subsequently utilizes a for loop and if statement to iterate the list, checking to see if there are IPs present in the remove list in the allow list. If any remove list IPs are discovered to be in the allow list, they will be removed before the file is updated using the .join() method to convert the data back into a string. To finalize the update, I used the .write() method.

Some alternatives to the code would be to implement the use of input() functions. By doing this automated manual input of IP addresses can be achieved. Alternatively, my code includes commented lines of importing another text file of addresses to be removed. If used, the manually input list must be commented out.

#importFile2 = "remove\_list.txt"  
#with open(importFile2, "r") as file2:  
# removeList = file2.read()  
#removeList = removeList.split()